IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR UNITED STATES PATENT

FOR

GRAPHICS PROCESSOR WITH TEXTURE MEMORY ALLOCATION SYSTEM

Inventors:

Stephen W. Edwards
77 Hartington Dr.
Madison, AL 35758

Attorney Docket: 1247/A38

Attorneys:
BROMBERG & SUNSTEIN LLP
125 Summer Street
Boston, MA 02110
(617) 443-9292

GRAPHICS PROCESSOR WITH TEXTURE MEMORY ALLOCATION SYSTEM

PRIORITY

This application claims priority from United States provisional patent application serial number 60/093,159, filed July 17, 1998, entitled "GRAPHICS PROCESSOR WITH TEXTURE MEMORY ALLOCATION SYSTEM" and bearing attorney docket number 1247/196, the disclosure of which is incorporated herein, in its entirety, by reference.

FIELD OF THE INVENTION

The invention generally relates to computer systems and, more particularly, the invention relates to processing graphics request data for display on a computer display device.

BACKGROUND OF THE INVENTION

Many conventional three dimensional graphics processing programs apply textures to graphical images to provide more realistic images. Textures typically are applied to graphical images by mapping a texture map to the graphical image. Texture maps typically are stored in volatile texture memory on a graphics accelerator and applied to graphical images by a local texture processor.

When a selected texture map is required, a program typically determines the location and type of texture map (*e.g.*, its dimensional type) in the texture memory. Once this information is determined, the program transmits a message to the texture processor with this information. Upon receipt of the message by the texture processor, the texture map is retrieved and applied to a graphical image of interest. Transmitting the message to the texture processor, however, requires bus bandwidth that preferably is utilized for transmitting other graphics request code.

In addition, texture memory commonly is configured as linear memory (*i.e.*, one dimensional). Many texture maps, however, are two and three dimensions. Storing a higher dimensioned texture map (*i.e.*, higher than one dimension) in linear texture memory thus often results in an inefficient allocation of memory resources. More particularly, many memory

locations undesirably are not used when storing higher dimensioned texture maps in linear texture memory.

SUMMARY OF THE INVENTION

In accordance with one aspect of the invention, a graphics accelerator for processing a graphical image includes a texture buffer for storing texture maps and data relating to the texture maps stored in the texture buffer, and a texture processor that performs texturing operations on the graphical image. The texture processor includes a fetching engine that retrieves texture packets, where each texture packet is stored in the texture buffer and associated with a texture map. Each texture packet includes data relating to the location of its associated texture map in the texture buffer.

In preferred embodiments of the invention, each texture packet is associated with a texture map that is different than the texture maps associated with any other texture packet in the texture buffer. Moreover, each texture packet may include data relating to the dimensional type of its associated texture map. Among other things, the dimensional type of each texture map may be one of a one dimensional texture map, a two dimensional texture map, and a three dimensional texture map.

In other embodiments, the texture processor further includes an input for receiving a texture message indicating that a texture map is to be utilized by the texture processor. The fetching engine retrieves selected texture packets from the texture buffer in response to receipt of the texture message. The texture processor also may include a parsing engine for parsing a fetched texture packet and determining information relating to the texture map associated with the fetched texture packet. In preferred embodiments, the information relates to the location in the texture buffer of the texture map associated with the fetched texture packet. The information also may relate to the number of dimensions of the texture map associated with the fetched texture packet.

In accordance with other aspects of the invention, a graphics accelerator for processing a graphical image includes a texture buffer for storing texture maps and data relating to the texture maps stored in the texture buffer, and a texture processor that performs texturing operations on the graphical image. The texture processor includes a texture packet generator that produces texture packets, where each texture packet is stored in the texture buffer and associated with a texture map. Each texture packet includes data relating to the dimensional type of its associated texture map.

In accordance with still other aspects of the invention, a graphics accelerator stores a texture map in linear texture memory of a graphics accelerator by first determining the dimension of the texture map, and then converting the texture map to a one dimensional texture map if the dimension of the texture map is determined to be more than one dimensional. The one dimensional texture map has a first number of consecutive data blocks. A second number of consecutive memory locations in the texture memory then are located, where the first number is equal to the second number. Once the consecutive memory locations are located, then the one dimensional texture map is stored in the located memory locations in the texture memory.

In preferred embodiments, a texture map is converted to a one dimensional texture map by first defining a plurality of data blocks within the texture map, and then assigning a sequence number to each of the data blocks. The sequence numbers preferably are consecutive numbers. In preferred embodiments, each consecutive data block of the one dimensional texture map is consecutively stored in the located memory locations.

In accordance with another aspect of the invention, an apparatus and method of applying a texture to a graphical image first locates a texture packet identifying the location of a texture map in a memory device. The texture packet then is parsed to determine the location of the texture map in the memory device. Once its location is determined, the texture map is retrieved and applied to the graphical image.

In accordance with still another aspect of the invention, a data structure is utilized to store data relating to a texture map. More particularly, the data structure includes a location field

identifying the location of the texture map in a memory device, and a dimension field identifying the dimension of the texture map.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects and advantages of the invention will be appreciated more fully from the following further description thereof with reference to the accompanying drawings wherein:

Figure 1 schematically shows a portion of an exemplary computer system on which preferred embodiments of the invention may be implemented.

Figure 2 schematically shows a preferred graphics accelerator that may be utilized in accord with preferred embodiments of the invention.

Figure 3 shows additional details of a preferred embodiment of a rasterization stage shown in figure 2.

Figure 4 shows a preferred process of storing texture maps in either of the texture memories shown in figure 3.

Figure 5 shows an exemplary two-dimensional texture map as it is converted into a one dimensional texture map.

Figure 6 shows a preferred method of retrieving a texture map from texture memory.

Figure 7 schematically shows a texture packet configured in accord with preferred embodiments of the invention.

DESCRIPTION OF PREFERRED EMBODIMENTS

Figure 1 shows a portion of an exemplary computer system 100 on which preferred embodiments of the invention may be implemented. More particularly, the computer system 100 includes a host processor 104 (*i.e.*, a central processing unit) for executing application level programs and system functions, volatile host memory 102 for short term data storage (*i.e.*,

random access memory), a graphics accelerator 106 for processing graphics request code in accord with preferred embodiments of the invention (see figure 4), and a bus coupling all of the prior noted elements of the system 100. The system 100 further includes a display device 108 for displaying the graphics request code processed by the accelerator 106. The graphics accelerator 106 preferably utilizes any well known graphics processing application program interface such as, for example, the OPENGL™ application program interface (available from Silicon Graphics, Inc. of Mountain View, California) for processing three dimensional ("3D") and two dimensional ("2D") graphical request code. In preferred embodiments, the host processor 104 executes a graphical drawing application program such as, for example, the PLANT DESIGN SYSTEM™ drawing program, available from Intergraph Corporation of Huntsville, Alabama.

Figure 2 shows several elements of the graphics accelerator 106. In preferred embodiments, the graphics accelerator 106 includes a double buffered frame buffer 200 (*i.e.*, having a back buffer and a front buffer) for storing the processed graphics request code in accord with the OPENGL™ interface. Among other things, the graphics accelerator 106 also preferably includes a geometry accelerator 202 for performing geometry operations that commonly are executed in graphics processing, a rasterizer 204 for rasterizing pixels on the display device 108, and a resolver 206 for storing data in the frame buffer 200 and transmitting data from the frame buffer 200 to the display device 108. As noted above, the graphics accelerator 106 preferably is adapted to process both 2D and 3D graphical data. For more information relating to preferred embodiments of the graphics accelerator 106, see, for example, copending United States patent application entitled "MULTI-PROCESSOR GRAPHICS ACCELERATOR", filed on even date herewith and naming Steven J. Heinrich, Stewart G. Carlton, Mark A. Mosley, Matthew E. Buckelew, Clifford A. Whitmore, Dale L. Kirkland, and James L. Deming. as inventors, the disclosure of which is incorporated herein, in its entirety, by reference.

Figure 3 shows additional details of a preferred embodiment of the rasterization stage 204 shown in figure 2. More particularly, the rasterization stage includes first and second rasterizers 300A and 300B that each deliver output data to the resolver stage 206. The first rasterizer 300A

includes a first texture processor 302A with accompanying first texture memory 304A (*i.e.*, buffer). In preferred embodiments, the first texture memory 304A is conventional one dimensional volatile memory. In a similar manner, the second rasterizer 300B includes a second texture processor 302B with accompanying second texture memory 304B. Like the first texture memory 304A, the second texture memory 304B also is conventional one dimensional volatile memory. The texture memory 304A and 304B may be any memory device that operates for such purposes, such as conventional synchronous dynamic random access memory (SDRAM). In preferred embodiments, each texture processor 302A and 302B may be an application specific integrated circuit ("ASIC") configured in accord with conventional processes. Each texture processor 302A and 302B further includes a fetching engine 308 for fetching specified data (discussed below) and a parsing engine 310 (also discussed below) that parses specified data.

In alternative embodiments (not shown), a single texture memory is utilized by each of the different texture processors 302A and 302B. The single texture memory may be partitioned and each partition may be assigned to a single texture processor. Alternatively, the texture map data may be arbitrarily stored in any location of the single texture memory by any of the texture processors on the graphics accelerator.

In accordance with preferred embodiments of the invention, texture maps of varying dimensions may be stored in either of the texture memories 304A and 304B. For example, texture maps of two dimensions, three dimensions, four dimensions and higher dimensions may be stored in either of the one dimensional texture memories. This enables either of the texture processors 302A and 302B to store data in their respective texture memories 304A and 304B in a more efficient manner.

Figure 4 shows a preferred process of storing texture maps in either of the texture memories. The first texture processor 302A and its accompanying first texture memory 304A are discussed relative to figure 4 by way of example. Accordingly, the process shown in figure 4 may be applied to the second texture processor 302B with its second texture memory 304B, or any other texture processor and accompanying texture memory.

The process begins at step 400 in which a texture map is retrieved by the host processor 104 (in response to instructions from a drawing or configuration program) from a library of texture maps in a nonvolatile memory. For example, such library may be stored on a hard disk in the computer system 100. The dimension of the texture map then may be determined by the host processor 104 (*e.g.*, by means of a driver application program executed by the host processor 104) and transmitted to the texture processor 302A with the texture map via a message (step 402).

In response to receipt of the texture map, the texture processor 302A responsively converts the texture map into a one dimensional texture map of one or more texture data blocks (step 404). In preferred embodiments, a texture data block ("block") includes data representing a maximum of 512 texels of a texture map.

When converting a two dimensional texture map, each of the blocks of data in the map may be assigned a consecutive sequence number on a block by block basis. This may be completed in any number of ways as known in the art. For example, as shown in figure 5, the top left texture data block may be assigned the first sequence number (*i.e.*, "1"), until the lower right block has the last sequence number (*i.e.*, "20"). Accordingly, each block in figure 5 is identified by a single address number and thus, is converted into a one dimensional map.

If a texture map is a three dimensional texture map, it may be converted into a two dimensional texture map by dividing it into a plurality of two dimensional texture map planes, and then converting each of the two dimensional map planes into one dimensional texture maps as described above. Higher dimensioned texture maps (*e.g.*, four dimensional texture maps) also may be converted into one dimensional texture maps by utilizing the above noted two and three dimensional conversion methods. It should be noted that other methods known in the art may be utilized.

The process then continues to step 406 in which a plurality of consecutive data blocks in the texture memory 304A are located. The preferably is done via software executing on the texture processor 302A. Of course, the located data blocks must equal at least the number of

resultant data blocks of the converted texture map. For example, at least twenty data consecutive data locations must be located in the texture memory 304A for storing the texture map shown in figure 5.

Once the appropriate data blocks are located, the process continues to step 408 in which the converted one dimensional texture maps are stored in the located data locations (*i.e.*, the appropriate blocks) in the texture memory 304A. In preferred embodiments, the texture processor 302A is configured via hardware to store the converted texture map in the texture memory 304A. A texture packet (discussed below and shown in figure 7) then is created at step 410 and stored in the texture memory 304A. In preferred embodiments, the location of the texture packet is stored in a packet look-up table that is utilized by the fetching engine 308 to retrieve such packet. Details of the packet retrieval process are discussed below.

Among other data, the texture packet may include the location of the stored converted texture map in the texture memory 304A, and the dimensional size of the texture map prior to being converted. For example, a texture packet may contain data indicating that the texture map shown in figure 5 is twenty data blocks long and stored in memory locations 1-20 in the texture memory 304A. Each texture packet in texture memory 304A preferably has a single unique texture map associated with it, and is stored in texture memory 304A in a location that is preconfigured to store texture packets. In some embodiments, the single texture map may include a family of mipmaps (discussed below).

It should be noted that the steps of the process shown in figure 4 may be preformed in orders other than that shown in the figure. For example, some steps may be performed simultaneously, while other steps may be performed either before or after each other.

Accordingly, this process enables the texture memory to store ~~texture memory to store~~ texture maps of different dimensions. For example, preferred embodiments of a given texture memory 304A can store texture data for one, two, three and four dimensional texture maps. To that end, the texture memory 304A preferably is logically structured as two banks of linear arrays of memory blocks. Each memory block has a size that corresponds to the size of the texture

blocks described above. Accordingly, each block includes 512 data locations that can store up to 512 texels of data. Any block may be located in the texture memory address space by using a map address format that contains a bank select bit, a block address field, and a displacement field. The bank select bit and block address field together form a linear address that points to a unique block within the texture memory 304A. The displacement field locates a given texel within the block. .

Each block in a group of blocks that encompass any texture map preferably is logically arranged to correspond with the dimension of the texture map being stored. The following list shows the texture space geometry for one dimensional, two dimensional, and three dimensional memory blocks. The terms depth, width, and height respectively refer to the well known texture space directions known in the art as "R", "S", and "T" :

One dimensional texture map: 512 (width) texels;
Two dimensional texture map: 16 (width) by 32 (height) texels; and
Three dimensional texture map: 8 (width) by 8 (height) by 8 (depth) texels.

Even though texture maps are divided into blocks, preferred embodiments permit each such block to be stored across two blocks in memory. Accordingly, the beginning of a block of a texture map may be identified at an offset within a memory block. Below are formulas for determining how many blocks of memory are required by a texture map:

<u>One dimensional texture map</u>

$N = (2^R + 2*B+D+511) / 512$

where the division is integral and:

N is the total number of blocks required;

R is the number of bits of resolution in the texture map;

B is one if the texture map has a border and zero if it has no border; and

D is the displacement of the first texel.

Two dimensional texture map

$$N = ((2^{SR} + 2*B+SD+15)/16) *((2^{TR} + 2*B+TD+31)/32)$$

where the division is integral and:

N is the total number of blocks required;

SR is the number of bits of resolution in the texture map in the S direction;

TR is the number of bits of resolution in the texture map in the T direction;

B is one if the texture map has a border and zero if it has no border;

SD is the S displacement of the first texel; and

TD is the T displacement of the first texel.

Three dimensional texture map

$$N = ((2^{SR} + 2*B+SD+7)/8) *((2^{TR} + 2*B+TD+7) /8)*((2^{RR} + 2*B+RD+7) /8)$$

where the division is integral and:

N is the total number of blocks required;

SR is the number of bits of resolution in the texture map in the S direction;

TR is the number of bits of resolution in the texture map in the T direction;

RR is the number of bits of resolution in the texture map in the R direction;

B is one if the texture map has a border and zero if it has no border;

SD is the S displacement of the first texel;

TD is the T displacement of the first texel; and

RD is the R displacement of the first texel.

Figure 6 shows a preferred method of retrieving a texture map from texture memory 304A. The process begins at step 600 in which a message from the host processor 106 is received requesting the texture map. The message may have originated from an executing drawing program executing on the host processor 106. In response to the message, the appropriate texture packet is retrieved from the texture memory 304A by the fetching engine 308 (*e.g.*, a driver) in the texture processor 302A (step 602). To that end, the fetching engine 308 accesses the packet look-up table to determine the location of the appropriate texture packet upon receipt of graphics request code requesting a specified texture map. As noted above, once located, the texture processor retrieves the appropriate texture packet for analysis.

The texture processor 302A then parses the data in the texture packet to determine the location of its associated texture map and/or the dimension of such map. Continuing with the above example, the texture packet associated with the texture map shown in figure 5 will include data bits indicating that the texture map is stored at memory locations 1-20, and that such texture map is two dimensional.

The texture map then is retrieved from the texture memory 304A by the texture processor 302A based upon the information in the texture packet (step 608). It then is determined at step 610 if the texture map is one dimensional. If it is a one dimensional texture map, then the process ends. If, however, it is determined that the texture map is not one dimensional, then the texture processor 302A reconstructs the texture map in accord with conventional processes (step 612). For example, the texture processor 302A may perform opposite operations to those executed when converting the texture map to a one dimensional map. Once the texture map is reconstructed and/or retrieved (as the case may be), it may be applied to a graphical image in accordance with conventional processes.

Figure 7 schematically shows a texture packet 700 configured in accord with preferred embodiments of the invention. Specifically, the texture packet shown in figure 7 includes ten rows of sixty-four bit words that each are preceded by a displacement field 702. Each sixty-four bit word is divided into high and low thirty-two bit words (*i.e.*, fields) that each can store specific

information. Accordingly, the texture packet includes a configuration field 704 that identifies the dimension of its associated texture map, a border color field 706 identifying the color of the border (if any) of the texture map associated with the texture packet 700, a level of detail clamps field 708 , and a plurality of map address fields 710 identifying the address to one texture map, or a set of mipmaps. In preferred embodiments, address data of the associated texture map only is stored in the first address field 710 if such map does not include a plurality of mipmaps. All other address fields 710 thus include no data. In the case where an associated texture map includes a set of mipmaps, the address of each of sixteen mipmaps may be stored in the sixteen address fields 710.

Preferred embodiments of the texture processors 302A and 302B include a plurality of texture registers that collectively define the state of the entire texturing system. Among those registers are a texture configuration register that specifies data relating to texture maps sizes, valid maps, borders, and texel format. A texture environment register specifies other information about a given texture, such as the blend environment, clamp mode, and a level of detail clamp enable. Both of these registers are loaded via a load texture registers request that loads all registers associated with the texture processor 302A. Additional texture registers specify other texturing information, such as the base address of all texture maps, level of detail calculation clamping, and the border and environment blend colors. These additional registers may be loaded with the above noted load texture registers request, or preferably as a group with a setup texture registers request.

Portions of the disclosed apparatus and method may be implemented as a computer program product for use with a computer system. Such implementation may include a series of computer instructions fixed either on a tangible medium, such as a computer readable medium (*e.g.*, a diskette, CD-ROM, ROM, or fixed disk) or transmittable to a computer system, via a modem or other interface device, such as a communications adapter connected to a network over a medium. The medium may be either a tangible medium (*e.g.*, optical or analog communications lines) or a medium implemented with wireless techniques (*e.g.*, microwave, infrared or other

transmission techniques). The series of computer instructions embodies all or part of the functionality previously described herein with respect to the system. Those skilled in the art should appreciate that such computer instructions can be written in a number of programming languages for use with many computer architectures or operating systems. Furthermore, such instructions may be stored in any memory device, such as semiconductor, magnetic, optical or other memory devices, and may be transmitted using any communications technology, such as optical, infrared, microwave, or other transmission technologies. It is expected that such a computer program product may be distributed as a removable medium with accompanying printed or electronic documentation (*e.g.*, shrink wrapped software), preloaded with a computer system (*e.g.*, on system ROM or fixed disk), or distributed from a server or electronic bulletin board over the network (*e.g.*, the Internet or World Wide Web). Of course, some embodiments of the invention may be implemented as a combination of both software (*e.g.*, a computer program product) and hardware. Still other embodiments of the invention are implemented as entirely hardware, or entirely software (*e.g.*, a computer program product).

Although various exemplary embodiments of the invention have been disclosed, it should be apparent to those skilled in the art that various changes and modifications can be made which will achieve some of the advantages of the invention without departing from the true scope of the invention. These and other obvious modifications are intended to be covered by the appended claims.